

# Fundamental Solutions for Water Wave Animation

CAMILLE SCHRECK, IST Austria  
CHRISTIAN HAFNER, IST Austria  
CHRIS WOJTAN, IST Austria

This paper investigates the use of fundamental solutions for animating detailed linear water surface waves. We first propose an analytical solution for efficiently animating circular ripples in closed form. We then show how to adapt the method of fundamental solutions (MFS) to create ambient waves interacting with complex obstacles. Subsequently, we present a novel wavelet-based discretization which outperforms the state of the art MFS approach for simulating time-varying water surface waves with moving obstacles. Our results feature high-resolution spatial details, interactions with complex boundaries, and large open ocean domains. Our method compares favorably with previous work as well as known analytical solutions. We also present comparisons between our method and real world examples.

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

Additional Key Words and Phrases: water waves, dispersion relation, Helmholtz equation, wave equation, fundamental solution, equivalent sources, interactive fluid simulation

## ACM Reference Format:

Camille Schreck, Christian Hafner, and Chris Wojtan. 2019. Fundamental Solutions for Water Wave Animation. *ACM Trans. Graph.* 38, 4, Article 130 (July 2019), 14 pages. <https://doi.org/10.1145/3306346.3323002>

## 1 INTRODUCTION

This paper presents new strategies for efficiently animating water surface waves, especially for simulation in large, open domains with abundant high-frequency visual details and detailed boundaries. Such scenarios are particularly challenging for the state of the art in computer animation, which either rely on finite-difference approximations [Tessendorf 2004a], require a grid or mesh for the entire domain [Jeschke et al. 2018; Jeschke and Wojtan 2015], or require Lagrangian wave samples to fill the domain wherever details are present [Jeschke and Wojtan 2017; Yuksel et al. 2007]. Many of these approaches become prohibitively expensive for open (practically infinite) domains with extremely high-frequency capillary ripples. Our method aims to simulate detailed water ripples down to the level of individual raindrops and pebbles on a shoreline in an infinite, open ocean.

This work achieves detailed water animations by deriving fundamental solutions to the linear equations for water surface waves. Previous works used fundamental solutions (also referred to as “Green’s functions”) to compute efficient animations of elastic deformations [James and Pai 1999] and fracture [Hahn and Wojtan 2015, 2016; Zhu et al. 2015], sound propagation [James et al. 2006; Mehra et al. 2013], fluid vortices [Brochu et al. 2012; Cottet et al. 2000; Pfaff et al. 2012; Weißmann and Pinkall 2010], soap films [Da et al.

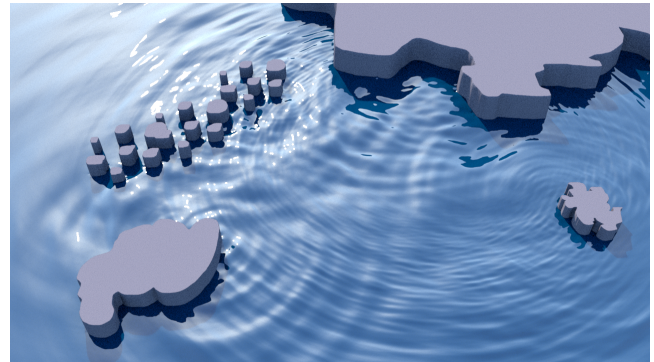


Fig. 1. We introduce the method of fundamental solutions to handle interactions between surface water waves with a large range of frequencies and complex detailed boundaries in an expansive open domain.

2015], three-dimensional liquids [Da et al. 2016; Keeler and Bridson 2014], keyframe animation [Barbič et al. 2012], and physically-based sculpting tools [De Goes and James 2017, 2018], to name a few. These fundamental solutions often enable unconditionally stable and extremely efficient analytical solutions for point sources, and linear combinations of them will produce mesh-free or boundary-only solutions (convenient for extremely large domains, and when high spatial frequencies make meshing difficult). However, to the best of our knowledge, our work is the first to investigate the use of fundamental solutions for animating water surface waves in Computer Graphics.

This work offers the following contributions to the state of the art in water animation:

- We introduce fundamental solutions for the animation of water surface waves in CG. Water waves have a frequency-dependent wave speed, so they present complications which are not solvable with previous techniques designed for linear elasticity and sound propagation. We introduce novel methods for handling this dispersive behavior.
- We apply analytical approximations to obtain a closed-form ripple function, which is useful for animating physically realistic rain drops in real-time.
- We introduce the use of the *method of fundamental solutions* for animating water waves, and we propose an interactive animation technique based on it. The resulting animation is not tied to a grid resolution, so we can animate infinite domains and zoom in arbitrarily close without degradation of visual detail.

Authors’ addresses: Camille Schreck, IST Austria; Christian Hafner, IST Austria; Chris Wojtan, IST Austria.

© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM. This is the author’s version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *ACM Transactions on Graphics*, <https://doi.org/10.1145/3306346.3323002>.

## 2 PREVIOUS WORKS

### 2.1 Method of fundamental solutions

This work aims to simulate the Navier-Stokes equations by first making a number of analytical calculations and approximations before eventually resorting to numerical techniques. We are not the first to seek a fundamental solution for water surface waves. Simple expressions for circular gravity-driven waves date back at least to Lord Kelvin [Thomson 1891]. More recently, Le Méhauté [1988] analyzed circularly symmetric ripples driven by both gravity and surface tension. He derived an approximate fundamental solution, which we leverage for the animation of an individual raindrop. But to our knowledge, the method of fundamental solutions has never been applied to water waves.

The *Method of Fundamental Solutions* (MFS), also known as the *Equivalent Sources Method* (ESM), proceeds by approximating a wave field by a set of point sources – outside the domain – which are analytic solutions of the wave equation. It is used in acoustics as an alternative to the Boundary Element Method, as it does not rely on a mesh and does not suffer from singularity problems at the boundaries. Lee [2017] reviews recent uses of ESM in computational acoustics.

In Computer Graphics, MFS has mainly been used for sound simulation. To speed up the computation of the sound radiation of a vibrating object, James et al. [2006] use fundamental solutions to approximate acoustic transfer functions for vibrating objects. Mehra et al. [Mehra and Manocha 2014; Mehra et al. 2013] use an ESM technique to compute the scattering of sound in a static environment. They represent both input sound waves (those arriving at an obstacle) and output waves (those scattered by an obstacle) as equivalent sources, and they employ pre-computed acoustic transfer matrices to model the relationships between the two types of wave sources.

### 2.2 Water surface waves in CG

Fully volumetric fluid solvers produce complex and beautiful non-linear fluid effects, but they can be prohibitively expensive, especially for interactive applications, or those which require highly detailed surface tension ripples. To get around the computational complexity of a 3d Navier-Stokes solver, many researchers have used the linear surface water wave approximation, which reduces the problem to a dispersive linear wave equation modeled as a 2d heightfield.

For ambient waves in a static environment, spectrum-based methods are common. These methods use a discretized Fourier transform to represent the waves as a sum of cosine waves [Hinsinger et al. 2002; Horvath 2015; Mastin et al. 1987; Tessendorf 2004b]. This approach is efficient for the open ocean, as it can have arbitrarily high spatial frequencies. Furthermore, it utilizes an analytic solution defined everywhere in time and space, enabling the use of arbitrary time steps and grid or mesh representations without the possibility of numerical instability (See [Hinsinger et al. 2002] for an example of an adaptive grid). However, handling complex boundary interactions can be quite difficult for these spectrum-based methods.

A popular way to animate interactive waves is to use local information on a grid to update the heightfield at each time step. Early

approaches in computer graphics discretize the linear shallow water equations [Kass and Miller 1990], which is a useful approximation for animating waves which have a much longer wavelength than the water depth. On the other end of the spectrum, several researchers investigate the simulation of deep water waves, which have an entirely different dispersive behavior. To animate deep water waves, a convolution kernel can be applied over the grid [Loviscach 2002; Tessendorf 2004a, 2014] to propagate waves, and boundaries can be handled by manipulating the wave height inside of obstacles. Canabal et al. [2016] use pyramid kernels and shadowed convolution to better deal with dispersion and boundaries, though the grid and kernels need to be fine enough to capture all boundary details. Geist et al. [2010] use a Lattice-Boltzmann method, which emulates wave behavior using 2d lattices. Thürey et al. [2006] also use a Lattice-Boltzmann method, but combines a small volumetric simulation (with 3d lattices) and a larger surface simulation (with 2d lattices). Irving et al. [Irving et al. 2006] also propose to improve the performance of 3d fluid simulation with 2d techniques by coarsening the simulation grid with tall cells away from the free surface interface. Recently, Jeschke et al. [2018] proposed a wavelet approach that convects a wavelet amplitude function on a grid. Their method can represent high frequency wave details even with a coarse grid, but their method sacrifices the ability to accurately keep track of coherent wave phases or reproduce common interference patterns.

A more global approach is used by Keeler and Bridson [2014] who solve the boundary integral equation on a 2d mesh. Jeschke and Wojtan [2015] produce highly detailed water waves by tracking wavefronts in a static environment, but they cannot handle interactions with moving obstacles. In both methods, the mesh needs to be fine enough around obstacles to accurately resolve boundary conditions.

Some methods employ Lagrangian particles to track water waves. Yuksel et al. [2007] track wave particles, each representing a wave crest for a certain wavelength. Jeschke and Wojtan [2017] proposed a similar approach with wave packets that rather represent a train of waves for a small range of frequencies.

Some methods also improve a coarse 3d fluid simulation by solving the wave equation on the animated surface (using a grid representation [Bojsen-Hansen and Wojtan 2013; Kim et al. 2013] or a Lagrangian representation [Bojsen-Hansen et al. 2012; Mercier et al. 2015; Thürey et al. 2010; Yang et al. 2016; Yu et al. 2012]) to add high-frequency turbulence.

This article proposes a different way to approach the water wave problem. Similar to the spectrum-based methods mentioned above, our fundamental solution method is independent of a grid or mesh, and the solution is defined on an infinite domain. However, we also gain wave interaction with obstacles. The accuracy of the boundary handling depends on a 1d sampling of sources and boundary points, rather than of the size of a 2d mesh. Thus our method can efficiently animate a large range of waves with detailed boundary conditions on an infinite open surface.



### 3 TECHNICAL BACKGROUND

#### 3.1 Fundamental Solutions

A *fundamental solution*  $F(\mathbf{x} - \mathbf{y})$  for a linear operator  $L$  satisfies the property

$$LF(\mathbf{x} - \mathbf{y}) = \delta(\mathbf{x} - \mathbf{y}).$$

If an analytical form of a fundamental solution can be found, then costly numerical integration can be completely avoided for certain problems. Analytic fundamental solutions can even have surprising new applications, like as a digital sculpting tool [De Goes and James 2017]. We can also combine multiple fundamental solutions—the superposition principle of linear partial differential equations (PDEs) allows us to solve PDEs with a different right-hand side

$$Lu(\mathbf{x}) = g(\mathbf{x}), \quad (1)$$

by applying a convolution to the fundamental solution:

$$u(\mathbf{x}) = \int_{\Omega} F(\mathbf{x} - \mathbf{y})g(\mathbf{y})d\mathbf{y}. \quad (2)$$

Thus, simply combining together multiple fundamental solutions allows us to solve many more problems than we started with. If  $g(\mathbf{x})$  is only non-zero on the boundary of the domain, then Equation 2 is equivalent to an integral over the boundary  $\partial\Omega$  instead of over the entire domain  $\Omega$

$$u(\mathbf{x}) = \int_{\partial\Omega} F(\mathbf{x} - \mathbf{y})g(\mathbf{y})d\mathbf{y}. \quad (3)$$

Numerically, this transformation reduces the computational complexity of approximating the integral by an order of magnitude ( $O(n^2) \rightarrow O(n)$  in 2d and  $O(n^3) \rightarrow O(n^2)$  in 3d). Direct numerical approximations of this boundary integral are called *Boundary Element Methods* [Sauter and Schwab 2010]. Boundary element methods typically run into practical difficulties when the fundamental solutions are singular at  $\mathbf{x} = \mathbf{y}$ . Researchers have proposed multiple approaches to circumventing this problem, like regularizing the fundamental solution [De Goes and James 2017], or devising an approximate boundary element method which places the fundamental solution sources slightly away from the query locations. These techniques, called the *Equivalent Sources Method* or the *Method of Fundamental Solutions* [Lee 2017], essentially approximate Equation 3 by replacing the integral with a finite sum of fundamental solutions, each centered at position  $\mathbf{y}_i$  and evaluated at position  $\mathbf{x}$ :

$$\tilde{u}(\mathbf{x}) = \sum_i^N a_i F(\mathbf{x} - \mathbf{y}_i)g(\mathbf{y}_i), \quad (4)$$

where  $\tilde{u}(\mathbf{x})$  is the approximation to the exact solution  $u(\mathbf{x})$ , and  $a_i$  is a quadrature weight based on the boundary sampling. More accurate solutions require more accurate boundary quadrature, typically resulting in more function evaluations. This summation must be evaluated at every point where we wish to know  $\tilde{u}(\mathbf{x})$ .

Solving for the variables  $a_i$  in Equation 4 typically requires the solution of a dense  $N$ -dimensional linear system, where  $N$  is the number of fundamental solution sources. Techniques like the fast multipole method [Coifman et al. 1993] and adaptive cross approximation [Kurz et al. 2002] can significantly reduce the computational complexity by speeding up matrix multiplication and removing the need to store the matrix explicitly.

Independent of its accuracy and computational complexity, the method of fundamental solutions has many desirable properties that are practically impossible to obtain with alternatives like the finite element method. For example, the approximate solution  $\tilde{u}(\mathbf{x})$  happens to be the *exact* solution to a perturbed PDE  $Lu = \tilde{g}(\mathbf{x})$ . Consequently, it will exactly satisfy many desirable properties defined by the linear operator—like if the true solution  $u(\mathbf{x})$  is harmonic, then the approximate solution  $\tilde{u}(\mathbf{x})$  will exactly satisfy the harmonic property as well. The approximate solution is also infinitely differentiable and has no numerical limit on its frequency or resolution. These properties are particularly useful in computer animation, because they prevent the solution from breaking down or exhibiting artifacts when the viewer gets too close to the scene.

#### 3.2 Linear Water Surface Waves

Like many previous works for simulating water waves, our method uses a linearized wave equation based on Airy wave theory:

$$\frac{\partial^2 u}{\partial t^2} = c_p^2 \nabla^2 u, \quad (5)$$

where  $u$  is a complex function and  $\text{Real}(u)$  is the water wave height,  $t$  is time,  $\nabla^2$  is the Laplacian operator, and  $c_p$  is the wavenumber-dependent phase speed. In this work, we use the phase speed for deep water, which assumes that the water is deeper than half a wavelength:

$$c_p(k) = \sqrt{\frac{g}{k} + \frac{\sigma k}{\rho}}, \quad (6)$$

where  $g$  is gravity,  $k$  is the wavenumber,  $\sigma$  is the surface tension strength, and  $\rho$  is the water density. Because the wave speed is different for different wavelengths, this is a *dispersive* wave equation. This wavenumber dependence makes the equation difficult to solve even numerically, as even simple finite difference methods require approximations to a dense linear operator [Tessendorf 2004a, 2014], and obtaining correct wave speeds with finite difference schemes is an active area of current research [Canabal et al. 2016]. Before we can find fundamental solutions for Equation 5, we first analyze the more familiar constant-speed wave equation.

#### 3.3 Wave equation

The constant-speed wave equation can be written

$$\left( \nabla^2 - \frac{1}{c^2} \frac{\partial^2}{\partial t^2} \right) u(\mathbf{x}, t) = 0, \quad (7)$$

where  $c$  is the constant wave speed. The constant-speed wave equation has a fundamental solution. For 2d surfaces this solution is:

$$F_t(\mathbf{x} - \mathbf{y}, t - s) = H(\tau - r/c) \frac{1}{2\pi} \frac{1}{\sqrt{\tau^2 - (r/c)^2}}, \quad (8)$$

where  $r = \|\mathbf{x} - \mathbf{y}\|$ ,  $\tau = t - s$  is the time since this wave began, and  $H$  is the Heaviside function.  $F_t$  is the solution of  $\left( \nabla^2 - \frac{1}{c^2} \frac{\partial^2}{\partial t^2} \right) F_t(\mathbf{x}, t) = \delta(\mathbf{x} - \mathbf{y})\delta(t - s)$ , which corresponds to one circular pulse sent from position  $\mathbf{y}$  at time  $s$ . As before, once we have a fundamental solution, we can write  $u(\mathbf{x}, t)$  as a convolution of sources. However,  $u(\mathbf{x}, t)$  is dependent on both space and time, so this convolution is a double

integral

$$u(\mathbf{x}, t) = \iint F_t(\mathbf{x} - \mathbf{y}, t - s) g(\mathbf{y}, s) d\mathbf{y} ds, \quad (9)$$

which is not analytically solvable in general. A common approach to simplifying this problem is to assume that the solution is separable in space and time, i.e.

$$u(\mathbf{x}, t) = p(\mathbf{x})T(t). \quad (10)$$

This separation allows us to split the solution into a system of two independent equations:

$$\nabla^2 p + k^2 p = 0, \quad (11)$$

$$\left( \frac{d^2}{dt^2} + \omega^2 \right) T = 0, \quad (12)$$

where  $k$  is the wavenumber, and  $\omega(k) = kc$  is the angular frequency of the wave. Equation 12 is a one-dimensional ODE describing the wave's temporal behavior. Its analytical solution is a harmonic oscillator with frequency  $\omega$ :  $T(t) = e^{-i\omega t}$ , where  $i = \sqrt{-1}$ . Equation 11 is a PDE known as the Helmholtz equation, which describes the spatial behavior of the wave. The Helmholtz equation has a fundamental solution

$$F_k(\mathbf{x} - \mathbf{y}, k) = -\frac{i}{4} H_0^{(2)}(k\|\mathbf{x} - \mathbf{y}\|), \quad (13)$$

where  $H_0^{(2)}$  is the 0<sup>th</sup> order Hankel function of the second kind, a highly-oscillatory complex function composed of radially-symmetric Bessel functions which decay with distance from the source.  $F_k$  solves  $\left( \nabla^2 - \frac{1}{c^2} \frac{\partial^2}{\partial t^2} \right) F_k(\mathbf{x}, t) = \delta(\mathbf{x} - \mathbf{y})e^{-i\omega t}$ , a wave equation with a constant oscillating source, instead of a time-dependent decaying pulse. It can be seen as the fundamental solution of the wave equation in the *frequency* domain, compared to  $F_t(\mathbf{x} - \mathbf{y}, t)$  which is the fundamental solution in the *time* domain.

Using this new fundamental solution, we write  $p(\mathbf{x}, k)$  as a convolution of sources

$$p(\mathbf{x}, k) = -\int \frac{i}{4} H_0^{(2)}(k\|\mathbf{x} - \mathbf{y}\|) g(\mathbf{y}, k) d\mathbf{y}. \quad (14)$$

The separability assumption is used in computer graphics for animating sound waves, but its applicability is limited because it cannot accurately capture more complicated time-dependent dynamics. As far as we are aware, the more general time-dependent fundamental solution  $F_t$  was only used in computer graphics to animate constant-speed elastic waves in 3d [De Goes and James 2018], where the behavior is simpler than 2d.

Unfortunately these fundamental solutions do not immediately solve our water wave animation problem. The dispersive wave equation is more complicated, because its wave speed  $c_p(k)$  varies with wavenumber. However, we note that a solution to the constant speed wave equation (7) is actually a valid solution to the dispersive wave equation (5) if its speed  $c$  obeys Equation 6, but it only describes the behavior of a single wavenumber  $k$ . Conveniently, due to superposition, we can combine these solutions for each wavenumber to get a complete solution to the dispersive wave equation:

$$u(\mathbf{x}, t) = \iint F_k(\mathbf{x} - \mathbf{y}, k) g(\mathbf{y}, k) e^{-i\omega t} d\mathbf{y} dk. \quad (15)$$

If we compare the original fundamental solution to the wave equation (8) to this one, we see that we do not need to integrate over time, but over wavenumber.

Our goal is to efficiently approximate the integrals in Equation 15 or Equation 9 in a manner that retains the arbitrarily high levels of detail, infinite smoothness, and general lack of visual artifacts as discussed in Section 3.1. We achieve this goal in two ways: we introduce a new closed-form function for the efficient animation of circular ripples in Section 4, and we solve the more general boundary problem using the method of fundamental solutions in Section 5.

## 4 CLOSED FORM CIRCULAR RIPPLES

As mentioned earlier, Le Méhauté [1988] derived an approximate fundamental solution for the motion of ripples driven by both gravity and surface tension forces. This section leverages his results with slight modification for the efficient animation of an individual raindrop. We discuss the approach in detail in the supplementary material [Hafner and Wojtan 2019] and summarize the key findings here.

### 4.1 Stationary-Phase Approximation

The integral in Equation 15 runs over two spatial dimensions and over wavenumbers, but we can reduce it to an integral over just wavenumbers by specializing the equation to the case of circular ripples emanating from a point  $\mathbf{y}_0$ . First, we change to polar coordinates and eliminate dependence on the angular coordinate by assuming radial symmetry, leaving an integral over the radius  $r = \|\mathbf{x} - \mathbf{y}_0\|$ . As explained in the supplementary material, to reduce the integral further, we make an analytically convenient choice of initial conditions that model the raindrop impact as an instantaneous disturbance in the velocity field around  $\mathbf{y}_0$ . This leaves us with the one-dimensional integral over wavenumbers

$$u(r, t) = -\int c_p(k) J_2(k) J_0(kr) \sin(\omega t) dk, \quad (16)$$

where  $J_0$  and  $J_2$  are 0<sup>th</sup> and 2<sup>nd</sup> order Bessel functions, and  $c_p(k)$  is the phase speed associated with wavenumber  $k$ .

Unfortunately, this integral is still not analytically solvable, and its highly oscillatory integrand requires many samples before numerical integration will converge.

Instead of relying on direct numerical approximation to make progress, we apply an analytical approximation technique known as the *method of stationary phase* (S-P) [Jeffreys and Jeffreys 1966]. The method relies on the observation that the high-frequency oscillations in integrals like this one tend to cancel out, so there is no need to pay attention to the function where it oscillates wildly. Instead, we focus the calculation on critical locations where oscillation is minimal, to capture the significant behavior.

These critical locations coincide with wavenumbers that render the phase of the oscillation stationary, i.e., where the group speed  $c_g = d\omega/dk$  attains a value of  $r/t$ . In Equation 16, there are at most two such wavenumbers:  $k_1$ , which corresponds to a low-frequency gravity wave, and  $k_2$ , a high-frequency capillary wave. These dominant wavenumbers depend on time  $t$  and distance  $r$ , and move at different speeds  $c_g$ , so we will see different wavelengths emerge and overlap as the waves ripple outwards.



Fig. 2. Closed form rain drop ripples animated in real time

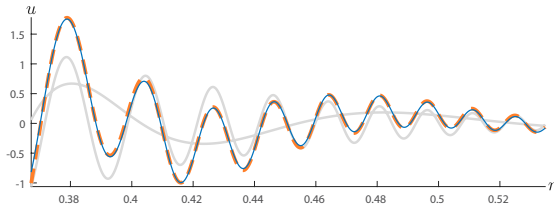


Fig. 3. Profile of a raindrop ripple two seconds after impact. Blue: S-P approximation to the surface height  $u$  as a function of the distance to the impact  $r$ . Orange: Ground truth. Gray: Decomposition into the gravity wave ( $k_1$ , low-frequency) and the capillary wave ( $k_2$ , high-frequency).

Applying this S-P approximation to Equation 16 replaces the integral with

$$u(r, t) \approx \frac{1}{\sqrt{rt}} \left( \frac{\omega(k_1)\sqrt{k_1}}{\sqrt{c_g'(k_1)}} \sin(\omega t - k_1 r) + \frac{\omega(k_2)\sqrt{k_2}}{\sqrt{c_g'(k_2)}} \cos(\omega t - k_2 r) \right), \quad (17)$$

where  $c_g'$  is the group speed derivative with respect to  $k$ . Figure 3 shows how the two-wave decomposition approximates the true solution. Details about the derivation of Equation 17 and a summary of formulae sufficient for implementation can be found in the supplementary material [2019].

## 4.2 Discussion and Implementation

We have effectively replaced expensive numerical integration with an efficient and numerically stable weighted sum of two sinusoids. Direct computation of Equation 17 is approximately as expensive as evaluating the original integrand twice. In Figure 4, we compare the accuracy of the S-P approximation with that obtained through numerical integration over wavenumbers. One second after impact, we need more than 60 quadrature samples to match the accuracy of the S-P approximation. Oscillations in the integrand of Equation 16 become higher-frequency as time increases, so four seconds after impact, 250 quadrature samples are required. For even higher timestamps, numerical integration becomes very unstable, as seen from the purple curve in Figure 4. Using the stationary-phase approximation to solve Equation 16 constitutes a marked increase both in stability and computational efficiency.

We pre-compute the dominant wavenumbers  $k_1$  and  $k_2$  as a function of the ratio  $r/t$  and store them in a one-dimensional look-up table for fast run-time performance. Our new ripple function is efficient to evaluate (only a constant factor slower than the most naïve approach of evaluating a single cosine wave), adds significant

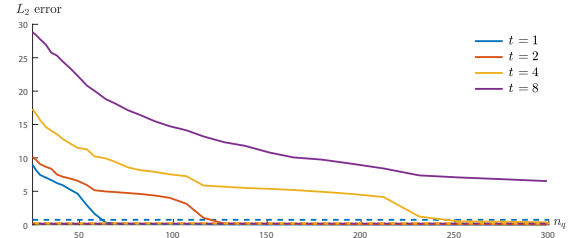


Fig. 4. Comparison of S-P approximation error with numerical quadrature error at different timestamps. The dashed lines show the error of the S-P approximation, and the continuous lines show the error of the quadrature solution as a function of the number of quadrature points  $n_q$ . The error is computed at different times after the impact:  $t = 1$ s (blue),  $t = 2$ s (red),  $t = 4$ s (yellow) and  $t = 8$ s (purple)

visual complexity by having wavelengths vary over space, and approximates the exact solution well. Furthermore, the expression is in closed form, so it will never blow up or exhibit visual artifacts. We incorporated this result into a real-time animation tool—the source code is available online. Figure 2 shows some examples of our real-time ripple animation tool in use.

## 5 WATER WAVES WITH BOUNDARY INTERACTIONS

The closed-form approximation described in the previous section works well for efficient simulation of isolated ripples like raindrops, but it does not incorporate interactions with boundaries. To address this problem, we turn to the method of fundamental solutions (MFS), which approximates Equation 15 or 9 by replacing its integrals with a discrete summation of point sources:

$$\tilde{u}(\mathbf{x}, t) = \sum_j a_j F_j(\mathbf{x}, t), \quad (18)$$

where  $\tilde{u}(\mathbf{x}, t)$  is the final numerical approximation to  $u(\mathbf{x}, t)$ ,  $F_j$  is a fundamental solution centered at position  $\mathbf{y}_j$ , and  $a_j$  is its relative complex weight. We can also consider a general input flow  $u_{\text{in}}$  provided by an artist or an existing simulation:

$$\tilde{u}(\mathbf{x}, t) = \sum_j a_j F_j(\mathbf{x}, t) + u_{\text{in}}(\mathbf{x}, t). \quad (19)$$

Each source  $F_j$  is a closed-form solution to our PDE, so any combination of  $a_j$  values will create physically realistic water motion. If the input waves  $u_{\text{in}}$  also approximate water wave behavior, then the entire solution will be physically realistic.

We can forbid waves from passing through solid objects by enforcing boundary conditions. Both Dirichlet  $u(\mathbf{x}, t) = 0$  and Neumann  $\nabla u(\mathbf{x}, t) \cdot \mathbf{n} = 0$  boundary conditions create interesting wave reflections ( $\mathbf{n}$  being the outward normal of the boundary at point  $\mathbf{x}$ ). We focus on Dirichlet boundary conditions in this section, but we can enforce Neumann conditions in a similar fashion. We satisfy the Dirichlet condition by setting  $\sum_j a_j F_j(\mathbf{x}, t) = -u_{\text{in}}(\mathbf{x}, t)$  for each point on the boundary and for each  $t$ —for Neumann conditions, it would be  $\sum_j a_j \nabla F_j(\mathbf{x}, t) \cdot \mathbf{n} = -\nabla u_{\text{in}}(\mathbf{x}, t) \cdot \mathbf{n}$ . This gives us a linear

system of equations  $M\mathbf{a} = -\mathbf{u}_{\text{in}}$ , or:

$$\begin{pmatrix} F_0(\mathbf{x}_0, t) & \dots & F_n(\mathbf{x}_0, t) \\ \vdots & & \vdots \\ F_0(\mathbf{x}_m, t) & \dots & F_n(\mathbf{x}_m, t) \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = - \begin{pmatrix} u_{\text{in}}(\mathbf{x}_0, t) \\ \vdots \\ u_{\text{in}}(\mathbf{x}_m, t) \end{pmatrix}, \quad (20)$$

where  $\mathbf{x}_0 \dots \mathbf{x}_m$  are the locations of boundary samples and  $a_0 \dots a_n$  are the complex weights for each point source. We discuss the position of the boundary samples and sources in Section 5.3 (see also Figure 6). Since we usually have more boundary samples than sources, to ensure a unique solution, we can solve the least squares system  $M^T M \mathbf{a} = -M^T \mathbf{u}_{\text{in}}$ , which finds the weights  $a_j$  that minimize the deviation from the boundary conditions in the least-squares sense.

After solving this linear system for the weights  $a_j$ , we can evaluate the solution anywhere in space and time with Equation 19. This evaluation is an easily parallelizable process which we perform on the GPU on a grid laid out in screen space (Similar to [Hinsinger et al. 2002; Jeschke et al. 2018]). Further speedups are possible with fast summation techniques like the fast multipole method, though we have not investigated this route.

### 5.1 Static Environment

Many detailed ocean environments have boundaries that do not change over time, like islands or buildings. Wave motion in such scenarios exhibits periodic wave behavior, and the wave spectrum will not change over time. We can exploit the static environment in these scenarios by solving the wave equation in the frequency domain once at the start of the simulation, and then playing back its periodic solution repeatedly for all times in the future. To do this, we first discretize the frequencies into a set of wave numbers  $k$ . The behavior of each wave number is independent from the others, so we decompose the solution into separate frequencies (essentially a discrete Fourier transform):

$$\tilde{u}(\mathbf{x}, t) = - \sum_k \tilde{u}(\mathbf{x}, k) e^{-i\omega_k t}. \quad (21)$$

Each  $\tilde{u}(\mathbf{x}, k)$  is a solution of the Helmholtz equation with  $\omega = c_p(k)k$ , so we use the fundamental solution defined in Equation 13 and solve the problem separately for each wave number:

$$\tilde{u}(\mathbf{x}, k) = \sum_j a_{jk} F_k(\mathbf{x} - \mathbf{y}_j, k) + u_{\text{in}}(\mathbf{x}, k) \quad (22)$$

$$= - \sum_j a_{jk} \frac{i}{4} H_0^{(2)}(k||\mathbf{x} - \mathbf{y}_j||) + u_{\text{in}}(\mathbf{x}, k). \quad (23)$$

We satisfy the Dirichlet boundary conditions by solving a linear system for the complex amplitudes  $a_{jk}$ :

$$\sum_j a_{jk} F_k(\mathbf{x}_b - \mathbf{y}_j, k) = -u_{\text{in}}(\mathbf{x}_b, k) \quad \forall \mathbf{x}_b, \quad (24)$$

where  $\mathbf{x}_{b=0 \dots m}$  are the boundary sample points. Note that this requires the Fourier transform of  $u_{\text{in}}$  into frequency components  $u_{\text{in}}(\mathbf{x}, t) = \sum_k u_{\text{in}}(\mathbf{x}, k) e^{-i\omega_k t}$ .

The amplitudes  $a_{jk}$  are independent of time, so we pre-compute them at the start of the simulation. At run time, the heights can be found at each frame by just adding the contribution of all the sources using Equations 21 and 22. In addition,  $u(\mathbf{x}, k)$  is also constant, so

we can pre-compute them as well if we know that the evaluation points  $\mathbf{x}$  will occur at fixed locations.

### 5.2 Varying environment

The solution outlined in the previous section works well for ambient waves like steady wind waves hitting a shore, but it cannot represent environments that change over time, like a boat wake or ripples caused by an object dropped in the water. Here we describe how to use the method of fundamental solutions for simulating water waves in an environment that changes over time.

**5.2.1 Time-domain solve.** One potential approach for allowing time-dependence is to use Equation 19 with the fundamental solution  $F_t$  defined in Equation 8, similar to how [Lee et al. 2010] computed time-dependent 3d acoustic waves. In this case, each spatial source  $F_j$  sends a pulse at every time step, and we have to compute the amplitudes for each source location as well as for every one of these pulses. The contribution of one source  $F_j$  at location  $\mathbf{y}_j$  is then:

$$F_j(\mathbf{x}, t) = \sum_{h=0}^{l-1} a_{jh} F_t(\mathbf{x} - \mathbf{y}_j, \tau_l - \tau_h), \quad (25)$$

where  $t = \tau_l = l \cdot \delta t$  is the time after  $l$  time steps of size  $\delta t$ , and  $\tau_h$  is the time when the  $h^{\text{th}}$  pulse was sent from this source.

We can then plug these  $F_j$  values into Equation 20 to solve for the pulse amplitudes at the current time step  $\tau_{l+1}$ . The amplitudes from previous pulses are already known, so we move their contributions to the right hand side and solve for the unknown  $a_{jl}$  weights:

$$\sum_j a_{jl} F_t(\mathbf{x}_b - \mathbf{y}_j, \delta t) = -u_{\text{in}}(\mathbf{x}_b, \tau_{l+1}) - \sum_j \sum_{h < \tau} a_{jh} F_t(\mathbf{x}_b - \mathbf{y}_j, \tau_{l+1} - \tau_h). \quad (26)$$

This approach has several drawbacks. First, the speed of the wave  $c$  used by Equation 8 does not depend on the wavenumber, so the waves are not dispersive. Second, the time needs to be discretized finely enough to capture each change of height and obtain smoothly oscillating waves. The Nyquist sampling theorem practically prohibits this method from simulating very high frequency wave details which are desirable in computer animation. Thirdly, the evaluation of the wave height at a point  $\mathbf{x}$  requires the summation of all previous pulses, so it is about an order of magnitude more costly to evaluate the surface:

$$\tilde{u}(\mathbf{x}, \tau_l) = \sum_j \sum_{h=0}^{l-1} a_{jh} F(\mathbf{x} - \mathbf{y}_j, \tau_l - \tau_h). \quad (27)$$

Finally, the fundamental solution in Equation 8 tends toward infinity for  $t \rightarrow r/c$  and needs to be regularized.

**5.2.2 Wavelet solution.** The frequency domain solution in Section 5.1 assumes that the wave height consists of periodic waves with a constant amplitude; it naturally handles arbitrarily high spatial frequencies, but it assumes the environment is fixed. In contrast, the time domain solution in Section 5.2.1 assumes the wave height is composed of a rapid succession of individual pulses; it naturally handles time-varying environments, but it can only compute high-frequency waves by emitting many different pulses one after another.



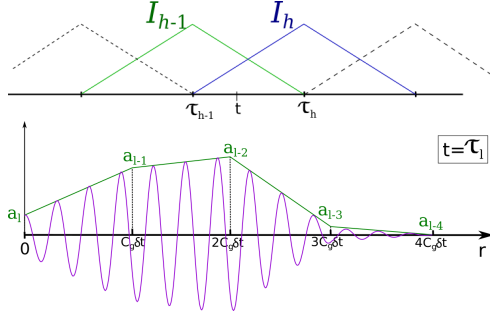


Fig. 5. The varying amplitude of a source is defined by discrete amplitudes defined at time  $\tau_h$  interpolated by tent functions (top). The interpolated amplitude defines an envelope for the wave (bottom figure shows a 1d spatial representation of the wave at time  $t = \tau_l$ ).

We seek a hybrid between these two approaches, which assumes that wave sources emit *oscillating waves with amplitudes that vary over time*. We achieve this by multiplying a time-varying amplitude with the harmonic fundamental solutions  $F_k$  from Equation 13:

$$F_j(\mathbf{x}, k, t) = a_j(t - \Delta t_{jk}(\mathbf{x})) F_k(\mathbf{x} - \mathbf{y}_j, k), \quad (28)$$

where  $\Delta t_{jk}(\mathbf{x}) = (\mathbf{x} - \mathbf{y}_j)/c_g(k)$  is the time taken for the wave energy to travel from  $\mathbf{y}_j$  to  $\mathbf{x}$ , and  $c_g(k) = \frac{\partial \omega}{\partial k}$  is the wave's group speed. We discretize the amplitude as a piecewise linear function over time:

$$a_j(t) = \sum_h I_h(t) a_{j,h}, \quad (29)$$

where  $a_{j,h}$  is the amplitude of the wave group emitted at time  $\tau_h$  from source  $j$ , and  $I_h$  is a tent function centered at  $\tau_h$  (see Figure 5). Similar to the time-domain solution, the source amplitudes  $a_{j,h}$  have to be recorded over time, but only two discrete amplitude values actually contribute to the height at a point  $\mathbf{x}$  (see Figure 5 top).

As described in Section 5.1, we decompose  $u$  into harmonic components to solve the problem separately for each frequency. For each of these frequencies, we solve Equation 20 at each time step, assuming that all the past amplitudes are already known and passing their contribution on the right side of Equation 20. Analogous to Equation 26,

$$\begin{aligned} \sum_j a_{j,l} I_l(\tau_l - \Delta t_{jkb}) F_k(\mathbf{x}_b - \mathbf{y}_j, k) &= -u_{\text{in}}(\mathbf{x}_b, t, k) \\ &- \sum_j \sum_{h < l} a_{j,h} I_h(\tau_h - \Delta t_{jkb}) F_k(\mathbf{x}_b - \mathbf{y}_j, k), \end{aligned} \quad (30)$$

where  $\Delta t_{jkb} = (\mathbf{x}_b - \mathbf{y}_j)/c_g(k)$  is the time it takes for the wave group to travel from  $\mathbf{y}_j$  to  $\mathbf{x}_b$ . We compute the final water surface height with:

$$\tilde{u}(\mathbf{x}, t) = \sum_k \sum_j F_j(\mathbf{x}, k, t) e^{-i\omega t}. \quad (31)$$

This new discretization has several advantages compared to the naïve one outlined in Section 5.2.1. First, due to the decomposition into harmonic components, the wave speed can vary for each frequency, so this method can successfully model dispersion. Second,

instead of summing over every emitted pulse throughout history, our piecewise linear amplitude discretization only needs to interpolate two of the previous amplitudes of each source. Third, this wavelet-style form of the fundamental solution allows us to take much larger time steps, because it only requires us to sample a slowly-varying wave amplitude signal, instead of a rapidly-varying wave height signal. At the same time, we gain the ability to animate waves with arbitrarily high spatial frequencies, because the frequency of  $F_k$  is now independent of time. This frequency decomposition is also exploitable computationally; we can choose a different time step size for each wave number  $k$  (our examples use  $\delta t_k = \frac{T_k}{4}$  or  $\delta t_k = \frac{T_k}{2}$  for wave period  $T_k = \frac{2\pi}{kc_p(k)}$ ), and we can sample the source and boundary points differently for each wavenumber as well. This frequency control allows us to devote fewer computational resources to motion with lower visual frequencies.

We can assume that the sources associated with distinct obstacles (like two different islands) can be computed independently from one another if they are far enough apart: in the case when all the contributions from the other obstacle can be put into  $u_{\text{in}}$ , the linear set of equations can be split up into smaller problems that are faster to solve.

It is also interesting to note that  $I_l(\tau_l - \Delta t_{jkb}) = I_0(-\Delta t_{jkb})$ . A first consequence of this is that the coefficients of the matrix do not depend on time but only on the distance between the sources and the boundary points. So the matrix can be pre-computed in scenarios with rigid obstacles. (where the distances between the sources and the boundary points do not change). Also, as  $I_0(-\Delta t_{jkb}) = 0$  when  $\Delta t_{jkb} > \delta t_k$ , the matrix is also sparse; each boundary point only directly influences close-by sources.

To model viscosity of the fluid, we multiply each fundamental solution  $F_j$  by a function  $d(\mathbf{x}, k) = e^{-\nu k^2 r/c_g}$  where  $\nu$  is a user-defined parameter that controls the viscosity of the fluid. This formula is the analytical solution for an ideal fluid with constant viscosity, but it neglects more complicated damping effects like surface contamination (please see [Le Méhauté 1988] for more details). When we incorporate this viscosity model into our wavelet MFS simulator, it causes waves to decay more quickly as they traverse the domain, eventually limiting each source's interaction with its distant neighbors. This effect gives us a computational advantage by reducing the number of terms in each summation. On the other hand, the viscosity model is surprisingly less useful for our static environment simulation method—although the system matrix becomes sparse, it seems to handle boundary conditions less accurately (witnessed by subtle waves leaking through boundaries). Consequently, we prefer to simulate inviscid water waves with our static-environment solution, similar to how spectrum-based approaches for deep water simulation omit viscosity entirely.

Although the static solution in Section 5.1 is unconditionally stable, the time-dependent solutions are not. Because each new amplitude is assigned based on previously computed amplitudes, it is possible for a positive feedback loop to grow out of control, particularly in a narrow channel where the sources are very close to one another and the fundamental solutions have very large values. One way to stabilize such simulations is to introduce viscosity as described in the previous paragraph. Another option is to damp

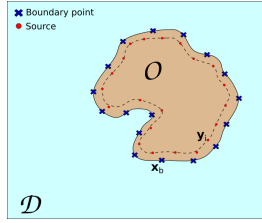


Fig. 6. The obstacles are represented by a set of boundary samples and a set of sources. They are sampled evenly, the boundary samples on the boundary and the sources on an offset curve inside the obstacle.

each source's change in amplitude by dividing it by some coefficient  $d_s$ : replacing the newly computed amplitude  $a_{il}$  by  $\tilde{a}_{il-1} = a_{il-1} + d_s(a_{il} - a_{il-1})$ . This heuristic damping prevents the amplitude from oscillating and diverging, but it may also reduce the accuracy by preventing the sources to react fast enough to a sudden change of in the input wave.

### 5.3 Implementation

We place the boundary sample points  $\mathbf{x}_b$  evenly along the solid boundaries, and we place the source points  $\mathbf{y}_j$  on an offset curve (as done in [Lee et al. 2010]) inside the obstacle (see Figure 6).

We adaptively sample the boundaries based on wavenumber, because the Nyquist theorem dictates that higher-frequency waves require denser boundary samples than lower-frequency ones. The examples in this paper use a boundary sampling density of  $\frac{6}{\lambda}$  and a source sampling density of  $\frac{3}{\lambda}$  (where  $\lambda = \frac{2\pi}{k}$  is the wavelength). Although this heuristic sampling scheme works well for our examples, it would be interesting to investigate a more principled boundary sampling technique, like the one proposed by James et al. [2006]. We discuss more about the placement of the sources and its effect on the stability and the results in Section 7.

The computation of  $\tilde{u}(\mathbf{x}, t)$  is an easily parallelizable process which we perform on the GPU on a grid laid out in screen space projected on the surface, similar to [Hinsinger et al. 2002]. Further speedups are possible with fast summation techniques like the fast multipole method, though we have not investigated this route. We use Eigen SVD to solve the linear systems in a least-squares sense. We believe the structure of this sparse symmetric positive definite linear system can be exploited to significantly speed up the performance, though we have not tried to optimize our solver performance.

## 6 RESULTS

Figure 2 shows a simple scenario with rapid animation of hundreds of raindrops using the closed-form ripple solver described in Section 4. Our MFS-based simulator was tested in various different scenarios, from simple examples with a single wavenumber to more complex examples with detailed boundaries and a large range of wavenumbers. We also compare our method to analytical solutions and real-life footage of water waves. Please see our supplemental video to view the animations described in this paper.

Alliney [1981] described an analytical solution for the diffraction of a planar water wave around a cylindrical object. We reproduce

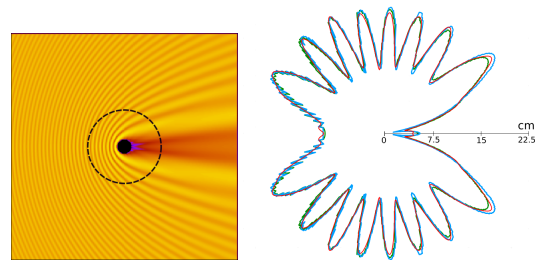


Fig. 7. Comparison with an analytical solution. A planar wave ( $\lambda = 1\text{m}$ ) meets a circular obstacle of radius  $1\text{m}$ . The left image shows the amplitude of the field  $\|u\|$ . The right image shows the amplitude of the spatial field at a distance  $d = 5\text{m}$  from the center (dashed circle) of the circle in every direction for the analytical solution (red), our steady-state simulator (blue) and our wavelet-based simulator (green).

this scenario and compare the magnitude of the complex wave field  $u(\mathbf{x})$  (represented in Figure 7, left) for a wave of wavelength  $\lambda = 1\text{m}$  diffracted by an obstacle of radius  $r = 1\text{m}$ . The graph in Figure 7, right, represents the magnitude of the field at a distance  $d = 5\text{m}$  away from the center of the obstacle in every direction. We compute this comparison both for our static (Section 5.1) and wavelet (Section 5.2.2) approaches, running the time-dependent wavelet simulation until it reaches a steady state. We can see that both the magnitude for the steady (blue) and wavelet (green) waves are close to the analytical solution (red). More importantly for computer graphics applications, they reproduce the same interference pattern.

Figure 8 shows how our MFS method reproduces common interference patterns with different boundary configurations. Notably, Figure 8(c) highlights the method's ability to compute shadows and diffraction around a thin wall, while Figure 8(d) shows how our method reproduces caustics where waves converge after reflecting from a concave obstacle. Figure 9 shows the interference pattern produced by a moving source. The vertically-translating source creates a curved shadow behind the circular obstacle.

Next, we compare our MFS method to recent related work for simulating water surface waves. Figure 10 shows how ours and other methods compute waves that interact with a single cylindrical obstacle. Our method (far left) accurately reproduces the analytical solution, as explained earlier. The wavefront tracking method of Jeschke & Wojtan [2015] (middle left) uses geometric optics to calculate the wavefronts, and they do not have an exact treatment of

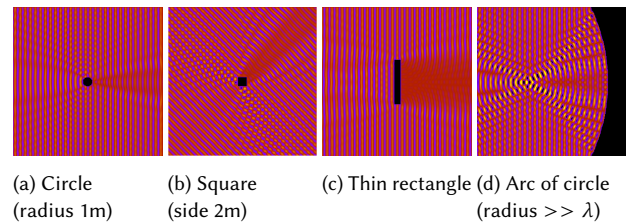


Fig. 8. Heightfield of a planar sinusoidal wave of wavelength  $\lambda = 1\text{m}$  reflecting on geometric obstacles

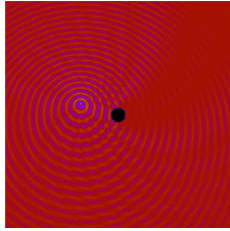


Fig. 9. A source of wavelength  $\lambda = 1\text{m}$  moving vertically reflecting on a cylindrical object of radius  $1\text{m}$ .

diffraction. Consequently, their method nicely aligns the wavefronts and reproduces an appropriate shadow behind the obstacle, but the inaccurate magnitude of the reflecting and diffracting waves does not produce the desired interference pattern. The Lagrangian wave packet simulator of Jeschke & Wojtan [2017] (middle right) is also based on geometric optics, so it successfully reproduces the shadow and has similar issues with diffraction. The method relies on point sampling to distribute packets over space, so the wavefronts are shifted a bit by sampling errors. Finally, we compare to the water surface wavelets approach of Jeschke et al. [2018] (far right). Using the parameters in their paper (a large number of spatial grid samples, but only a small number of wave direction samples) causes the wave angles to exhibit discretization artifacts. The method also exhibits damping due to semi-Lagrangian advection and an artificial angular diffusion term. Figure 11 compares to this method in a more complex scenario. Here, the method’s artificial angular diffusion still causes wave travel directions to mix, removing much of the directional bias that should be caused by the islands. This method will also have fundamental difficulties producing plausible diffraction patterns, because its underlying geometric optics theory cannot accurately model diffraction, and because it simulates only 1–4 representative wavelengths. On the other hand, the method of Jeschke et al. [2018] is significantly more efficient than ours when animating this scenario. Their method can reach real time while ours still takes 2 or 3 seconds per frame for this example.

Figure 12 shows how our wavelet-based MFS approach handles a more complex scene with a large range of frequencies. This animation features wavelengths ranging from 5m to 3mm and obstacles of size 30m to 3cm. The corresponding video zooms to show smaller and smaller drops falling into the water. Figure 13 shows several boat wakes produced by emitting waves from moving sources interacting with detailed boundary geometry. Note the frequency-dependent behaviors as waves reflect and diffract along the coastline; longer wavelengths ignore smaller coastline features, while shorter wavelengths scatter based on the high-frequency geometric details. Figure 14 shows an animation with a large number of wave sources interacting with obstacles.

### 6.1 Computation time

The closed-form circular ripples animation described in Section 4 takes very little time to compute. There is no interaction between drops, and the only numerically difficult task of finding roots for a large number of wavenumbers only takes place in one dimension,

Table 1. Performance of our wavelet MFS approach based on the number of fundamental solutions and boundary samples, for one frequency.

$n_{\text{sources}}$	$n_{\text{boundaries}}$	$t_{\text{solve}}(\text{sec})$	$t_{\text{sum}}(\text{sec})$
100	400	0.008	0.06
1000	4000	0.7	0.3
5000	20000	5.6	1.5
10 000	40000	21	3
50 000	200000	-	15

and it is done as a pre-computation. The only computationally costly operations at run time are GPU summation of wave heights for each grid point, which is the same for any simulation method based on Fourier transforms, wave packets, or wavelets. We can easily compute hundreds of rain drops while rendering at real-time rates (60fps) and did not stress test the method beyond this point.

Our MFS method is significantly more computationally intensive, because it handles significantly more general wave interactions. For each wavenumber in our MFS simulation, the two main steps of our algorithm are: (step 1) solving the linear system to find the amplitude of the sources, and (step 2) summing all the sources for each point on the display grid. In the following discussion of computational complexity, we will call the number of sources for each wavenumber  $n_{\text{sources}}$ , the number of boundary points for each wavenumber  $n_{\text{boundaries}}$ , and the fixed number of grid points for wave height evaluation  $n_{\text{grid}}$ .

Step 1 requires solving a  $n_{\text{boundaries}} \times n_{\text{sources}}$  linear system in the least-squares sense. For waves in a static environment, this step is only done once at the beginning of the simulation, and it requires a few minutes of pre-computation. For our wavelet-based MFS method, we compute step 1 every time step  $\delta t$ . We usually use  $\delta t = \frac{T}{4}$  where  $T$  is the period corresponding to the wavenumber  $k$ .  $\delta t$  is usually smaller than the time between each displayed animation frame. The matrix of the system is constant, so we precompute its SVD decomposition. The time  $t_{\text{solve}}$  shown in Table 1 is the time taken to find the new amplitudes of the sources using this decomposition for a system of size  $n_{\text{boundaries}} \times n_{\text{sources}}$ . For the time-dependent solver, which has a sparser matrix consisting of many disconnected components, we subdivide the system into smaller independent problems. We compute the amplitudes in a separate thread in parallel to the computation of step 2 and the rendering.

Step 2 is of complexity  $n_{\text{sources}} \times n_{\text{grid}}$ , but the computation can be done in parallel for each grid point. This step computes the displayed grid representing the surface of the water, and it has to be performed once for each displayed frame.  $t_{\text{sum}}$  in Table 1 is the time needed for this step using a simple GPU implementation according to the number of sources on a  $300 \times 300$  grid. Table 2 shows statistics for the same scene with different sampling densities and grid resolutions, and Figure 15 shows snapshots of these animations.

### 6.2 Comparison with real world

We compared the results from our simulations with a few real-world examples. Figure 16, left, shows video footage of a real lake shore reflecting and diffracting wind waves. The many small ripples are caused by non-linear phenomena but we are able to create a similar



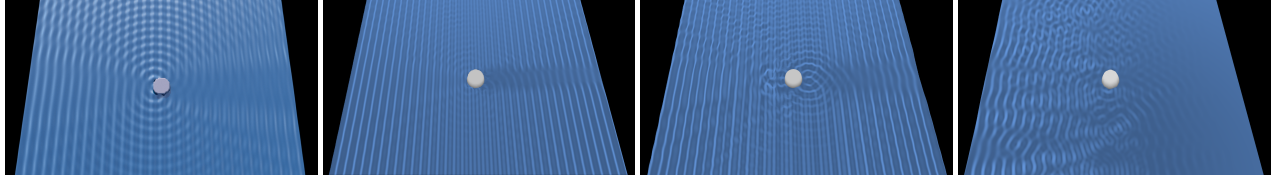


Fig. 10. A 1-meter plane wave interacting with a cylinder. Our method (left), wavefront tracking [Jeschke and Wojtan 2015] (middle left), wave packets [Jeschke and Wojtan 2017], water surface wavelets [Jeschke et al. 2018]

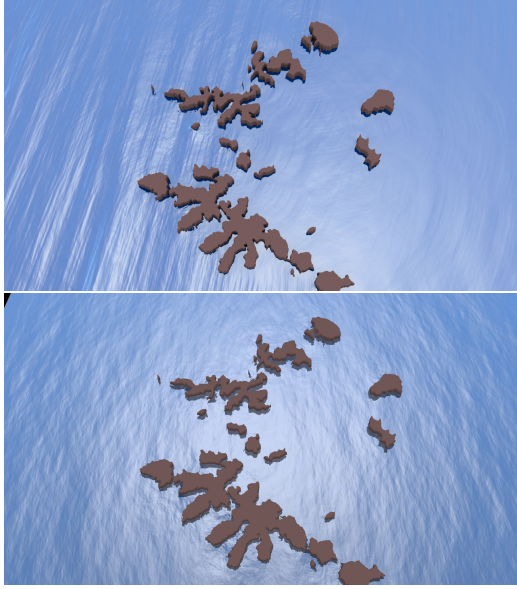


Fig. 11. Comparison between the Water surface wavelets [Jeschke et al. 2018] (top) and our method (bottom) in a more complex case. Spectrum of frequencies ranging from 0.05 to 10m.

Table 2. Timing information for the scene in Figure 15 for different resolutions of frequencies and grid. The linear system is not solved each frame ( $dt = 0.025$ ) but each time step  $\delta t$ .  $n_{s\_total}$  represents the total number of sources including all the frequencies.

figure	$n_{freq}$	$n_{s\_total}$	$\delta t$	grid size	$t_{sum}$	$t_{solve}$
(a)	23	56000	$9 * dt$	$600 \times 600$	9s	4s
(b)	12	9000	$22 * dt$	$300 \times 300$	1.8s	2s
(c)	4	1700	$32 * dt$	$300 \times 300$	0.35s	0.3s
(d)	9	5000	$32 * dt$	$150 \times 150$	0.2s	0.6s

effect by showing only the reflection of high-frequency planar waves in addition to the larger wind waves (Figure 16, right). Figure 17, left, shows waves emanating from a point source, reflecting off a circular boundary, and focusing in a small region before separating and focusing again. We compare this animation with video footage from a droplet falling into a tea cup (Figure 17, right).

## 7 DISCUSSION AND LIMITATIONS

We have introduced various techniques for applying the method of fundamental solutions to animate water waves, and each one has its own strengths and weaknesses, which we will elaborate upon in this section.

### 7.1 Source placement, stability, and time step

The static solution described in Section 5.1 is unconditionally stable. Although the method will always produce an animation, its accuracy and plausibility will still be affected by the number or locations of source points or boundary points. Insufficiently dense boundary and source sampling can cause waves to scatter inaccurately, resulting in some missing boundary details or perhaps some waves passing through obstacles. To visualize the effect of an inaccurate solution, Figure 19 shows the error of the least-squares solution (Equation 20) for a planar wave ( $\lambda = 1$ m) meeting a long, thin obstacle, with sources spaced apart by the distance  $d_{sample}$  along the obstacle border. We use the error metric:  $err = \frac{\|Ma + u_{in}\|_1}{\|u_{in}\|_1}$ . The three visualizations on the plot represent the result of the simulation for  $d_{sample} = 0.1, 0.5$ , and  $0.9$ . Similarly, Figure 18 shows a planar wavefront reflecting off of a circular obstacle with varying sampling density. The middle plot shows the amplitude at 5 meters from the center of the obstacle according to each sampled direction. Adding more sources makes the simulation converge closer to the analytical solution (see Figure 18 (bottom)). Figure 18 (top) compares the heightfield for a poor sampling (left, density of  $2/\lambda$  corresponding to the orange curve) and one closer to the analytical solution (right,  $5/\lambda$ , blue curve).

In contrast to the unconditionally stable static solution, the stability of the time-varying wavelet solution (Section 5.2.2) can be influenced by the placement of the sources and the time-step size. During each iteration, the simulation solves for the new amplitudes of waves emitted over the course of one time step  $\delta t$ . Thus, each source amplitude depends on the boundary samples within a radius  $c_g \delta t$ . If the time step is extremely large, then the amplitude is determined by all boundary points in the entire domain. This limiting case is similar to the static solution described above, and just as stable. On the other hand, if the time step is extremely small, the wave cannot propagate far enough to interact with any boundary points. This limiting case has no well-defined solution and is unstable.

For moderate time-step sizes, the wave amplitudes are determined by their small local neighborhood. Afterward, this new wave amplitude eventually interacts with boundary points further away and influences new amplitudes. In this way, numerical errors from one source can propagate to another. Numerical instabilities may



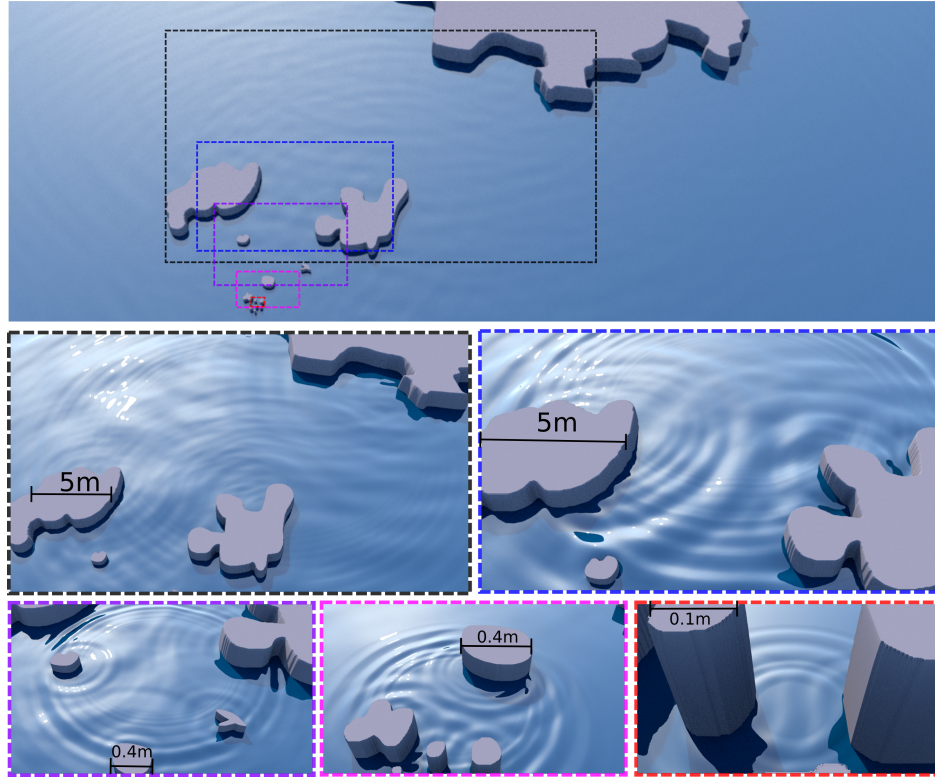


Fig. 12. Our method is able to exhibit many orders of magnitude of geometric levels of detail in the same simulation. Each image is a zoomed-in version of the previous one.

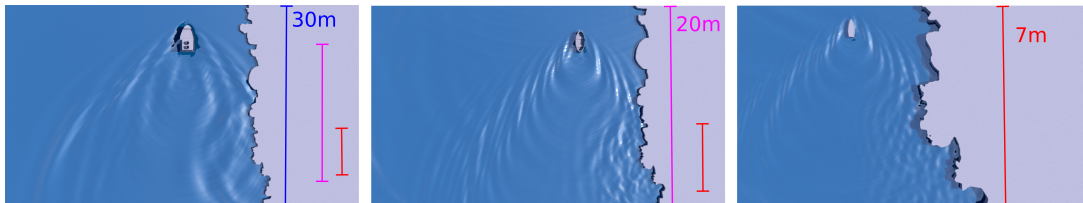


Fig. 13. Boat wakes of various wavelengths interacting with a detailed coastline.

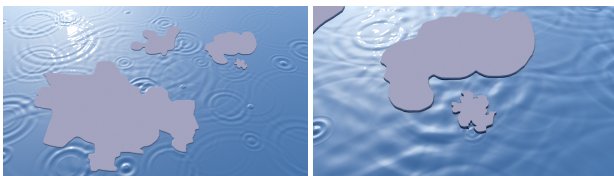


Fig. 14. Rain drops interacting with obstacles ranging in size from 50cm to 10cm

result if these errors lead to a systematic overestimation of wave amplitude at each interaction. Conveniently, wave amplitudes (and their associated numerical errors) naturally decay in our simulation, due to both viscosity and geometric spreading over large distances. Ideally, the time step, offset of the sources, and density should be chosen such that each source would have several boundary samples

close enough and inversely, so that the matrix  $M$  (see Equation 30) would be well-conditioned. Note that, while a large *simulation* time step might be required for stability, the time step for displaying the simulation can be much smaller.

This stability behavior is in direct contrast to explicit wave equation solvers which require small time steps to remain stable—larger time steps cause more sources to communicate within a single timestep, making our method behave like an implicit solver, with the limiting behavior as  $\delta t \rightarrow \infty$  being the time-independent solution discussed in Section 5.1. However, very large time steps in a quickly changing environment can cause changes of amplitude to be overlooked or waves to pass through obstacles.

## 7.2 Neumann vs. Dirichlet boundary conditions

Neumann boundary conditions more accurately model water waves reflecting off a boundary, compared to Dirichlet conditions, which

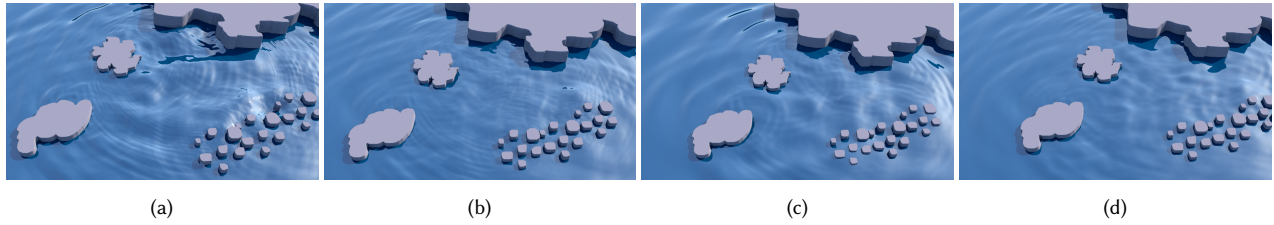


Fig. 15. The same simulations with varying numerical sampling parameters and grid resolutions, corresponding to Table 2.

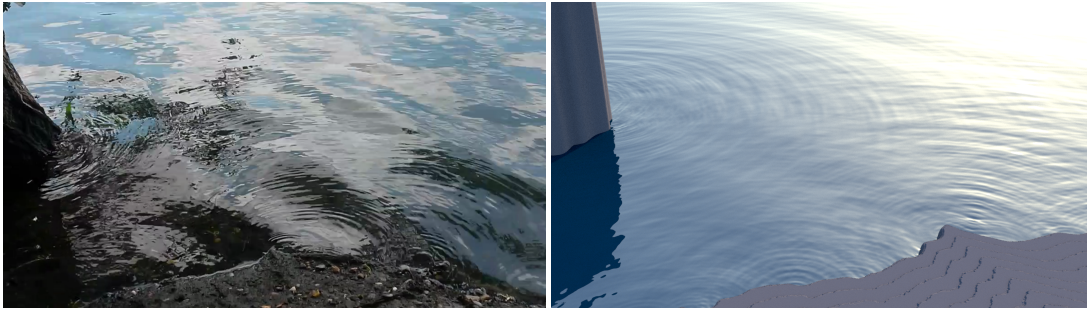


Fig. 16. Video footage of a lake scattering high-frequency waves (left) and a similar effect created by our simulator (right).



Fig. 17. Video footage of a drop in a tea bowl (of radius 5cm) (left). A similar animation is reproduced with our method (right).

are more appropriate for elastic waves with the medium pinned at the boundary. Using a Dirichlet condition instead of a Neumann will lead to the following specific inaccuracies: First, reflecting waves will be inverted (or, equivalently, shifted by a  $\pi$  phase offset) for Dirichlet conditions compared to Neumann ones. Second, because Dirichlet conditions pin the surface to the boundary, the resulting animations have weaker diffraction effects (which require waves to travel tangent to the boundary) (see Figure 20).

Despite these inaccuracies, we found it difficult to tell the difference between the boundary conditions in more complicated scenarios. (For example, see Figure 21, which illustrates two complicated island environments with different boundary conditions.) Dirichlet conditions are also slightly less expensive to compute and easier to implement (they compute function values instead of normal derivatives at boundaries). They also tend to be more numerically stable, likely because the Dirichlet fundamental solution is singular at the boundary ( $\propto 1/r$ ), while the Neumann fundamental solution is hypersingular ( $\propto 1/r^2$ ).

### 7.3 Limitations and future work

The method proposed in this paper was derived by placing several strong assumptions on the equations for fluid flow. First, the theory assumes small wave amplitudes, so this method is not suitable for simulating large overturning waves or splashes. Second, the theory only admits such a convenient fundamental solution if the water depth is constant. We employ the *deep water* assumption in this paper, which holds whenever the water depth is significantly larger than the wavelength of the water surface waves. Thus, this approach will not be able to animate phenomena caused by varying water depths, like refracting waves and wave shoaling effects.

Our closed-form solution for animating circular water ripples (Section 4) gains its computational efficiency and stability by assuming that the wave field is radially symmetric. Consequently, it will not accurately animate waves in a spatially varying environment (which would encourage refraction), and it cannot simulate waves interacting with obstacles. Our method for animating waves in a static environment (Section 5.1) gains its simplicity from an assumption that the environment is fixed; its beneficial computational



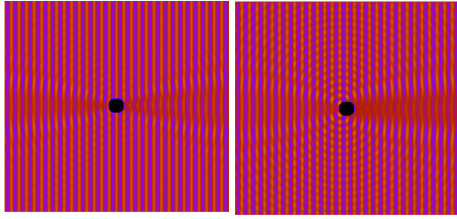


Fig. 18. Comparison between different sampling densities for the same simulation of a planar wave ( $\lambda = 1\text{m}$ ) meeting a circular obstacle of radius 1m. On the middle, we perform the same test as Figure 7 with different sampling densities per unit wavelength:  $5/\lambda$  (blue, 22 sources),  $3/\lambda$  (green, 15 sources),  $2.5/\lambda$  (red, 11 sources), and  $2/\lambda$  (orange, 9 sources). The bottom figure shows the (normalized  $L_1$ ) error of our simulation compared to the analytical solution according to the number of sources.

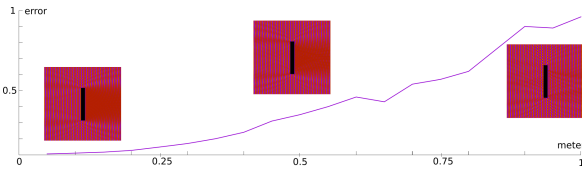


Fig. 19. Error of the least-squares solve according to the distance between sources along the border. A planar wave ( $\lambda = 1\text{m}$ ) meets a long thin obstacle. The pictures on the graph represent from left to right a distance of 0.1 (density  $10/\lambda$ ), 0.5 (density  $2/\lambda$ ) and 0.9 (density  $\approx 1.1/\lambda$ ).

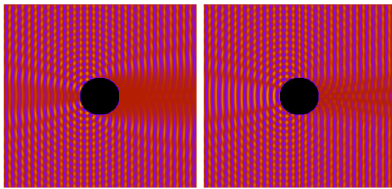


Fig. 20. Comparison between Dirichlet (left) and Neumann (right) boundary condition: A linear wave ( $\lambda = 1\text{m}$ ) meets a circular obstacle of radius 1m (top) and one of radius 3m (bottom).

properties (unconditional stability and a single pre-computation with efficient wave evaluation thereafter) will not be realizable in the presence of time-varying input waves. The method discussed in Section 5.2.2 is the most general technique in this paper, but it

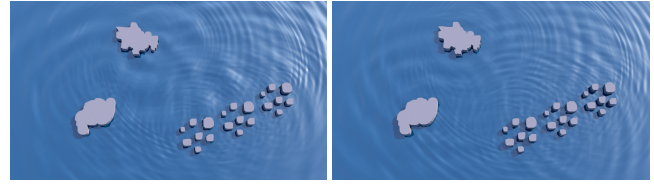


Fig. 21. Comparison Neumann-Dirichlet in a more complex example.

is not as computationally efficient as the other two. This technique requires a linear system solve at each time step, and its numerical stability is conditioned on numerical parameters like the time step size and the placement of source and boundary points.

The methods described in this paper exhibit a clear trade-off between generality and computational efficiency. The circular ripple method is extremely simple and can animate hundreds of waves faster than real-time, but it cannot animate interactions with obstacles. The static simulator requires a substantial pre-computation step and can successfully animate waves reflecting off obstacles, but the environment must be static. The wavelet approach falls short of real-time animations due to its complexity, but it can animate waves with moving obstacles and time-varying sources. The solution of a linear system each time step makes this method significantly more expensive than other real-time wave animation techniques, like methods based on explicitly integrated PDEs or particles. Consequently, our wavelet technique will not be the best method to use if the animation does not require large open domains or highly resolved wave details. In the future, we may be able to speed up our method by optimizing code, taking better advantage of GPU acceleration, and exploiting the sparse SPD structure of our linear system. These methods should also see significant speedup with fast summation techniques like the fast multipole method.

## 8 CONCLUSIONS

In this article, we introduced the use of fundamental solutions for animating water surface waves. We discussed a variety of different simulation strategies for efficiently simulating closed-form raindrops, for computing interaction of steady ambient wave with obstacle, and for computing more complicated water wave interactions with time-dependent boundaries.

The method reproduces physically accurate interference patterns. Our un-optimized research implementation creates detailed scenes with high frequency waves and boundaries at a few frames per second and achieves interactive rates at lower resolutions. Due to the convolutional nature of fundamental solutions, our low resolution simulations provide a remarkably accurate preview of higher resolution simulations with the same approach.

Our method is not as efficient as some alternative methods for animating simple water waves, but it can better handle complex boundaries with small features and simulate infinitely large domains.

We believe our closed-form water ripples could also help improve the realism of full 3d Navier-Stokes simulations by adding high-resolution ripples around spray and small droplets, so we would like to investigate what changes would be necessary to approximate

circular ripples on a moving surface. Our water wave fundamental solutions may also be useful for applications like animation compression and detail enhancement—an existing wave simulation could be approximated by a convolution of fundamental solutions, stored in a very sparse manner, and played back at arbitrarily high resolutions in unbounded domains.

## ACKNOWLEDGMENTS

We wish to thank the anonymous reviewers and the members of the Visual Computing Group at IST Austria for their valuable feedback. This research was supported by the Scientific Service Units (SSU) of IST Austria through resources provided by Scientific Computing. We would also like to thank Stefan Jeschke for providing the videos and pictures for the comparisons with previous works.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No. 638176 and No. 715767, and Marie Skłodowska-Curie grant agreement No. 665385.



## REFERENCES

- Stefano Alliney. 1981. Water waves diffraction around cylindrical obstacles. *Applied Mathematical Modelling* 5, 4 (1981), 237–240.
- Jernej Barbič, Funshing Sin, and Eitan Grinspun. 2012. Interactive editing of deformable simulations. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 70.
- Morten Bojsen-Hansen, Hao Li, and Chris Wojtan. 2012. Tracking surfaces with evolving topology. *ACM Trans. Graph.* 31, 4 (2012), 53–1.
- Morten Bojsen-Hansen and Chris Wojtan. 2013. Liquid surface tracking with error compensation. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 68.
- Tyson Brochu, Todd Keeler, and Robert Bridson. 2012. Linear-time smoke animation with vortex sheet meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 87–95.
- José A. Canabal, David Miraut, Nils Thürey, Theodore Kim, Javier Portilla, and Miguel A. Otaduy. 2016. Dispersion Kernels for Water Wave Simulation. *ACM Trans. Graph.* 35, 6, Article 202 (Nov. 2016), 10 pages.
- Ronald Coifman, Vladimir Rokhlin, and Stephen Wandzura. 1993. The fast multipole method for the wave equation: A pedestrian prescription. *IEEE Antennas and Propagation Magazine* 35, 3 (1993), 7–12.
- Georges-Henri Cottet, Petros D. Koumoutsakos, D. Petros, et al. 2000. *Vortex methods: theory and practice*. Cambridge university press.
- Fang Da, Christopher Batty, Chris Wojtan, and Eitan Grinspun. 2015. Double bubbles sans toil and trouble: discrete circulation-preserving vortex sheets for soap films and foams. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 149.
- Fang Da, David Hahn, Christopher Batty, Chris Wojtan, and Eitan Grinspun. 2016. Surface-only liquids. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 78.
- Fernando De Goes and Doug L. James. 2017. Regularized kelinlets: sculpting brushes based on fundamental solutions of elasticity. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 40.
- Fernando De Goes and Doug L. James. 2018. Dynamic kelinlets: secondary motions based on fundamental solutions of elastodynamics. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 81.
- Robert Geist, Christopher Corsi, Jerry Tessendorf, and James Westall. 2010. Lattice-boltzmann water waves. In *International Symposium on Visual Computing*. Springer.
- Christian Hafner and Chris Wojtan. 2019. Supplementary Material: Closed Form Integration of Gravity-Capillary Rings. (2019).
- David Hahn and Chris Wojtan. 2015. High-resolution brittle fracture simulation with boundary elements. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 151.
- David Hahn and Chris Wojtan. 2016. Fast approximations for boundary element based brittle fracture simulation. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 104.
- Damien Hsingier, Fabrice Neyret, and Marie-Paule Cani. 2002. Interactive Animation of Ocean Waves. In *ACM-SIGGRAPH/EG Symposium on Computer Animation (SCA'02)*, Stephen N. Spencer (Ed.). ACM SIGGRAPH, San Antonio, United States, 161–166.
- Christopher J. Horvath. 2015. Empirical Directional Wave Spectra for Computer Graphics. In *Proceedings of the 2015 Symposium on Digital Production (DigiPro '15)*. ACM.
- Geoffrey Irving, Eran Guendelman, Frank Losasso, and Ronald Fedkiw. 2006. Efficient Simulation of Large Bodies of Water by Coupling Two and Three Dimensional Techniques. *ACM Trans. Graph.* 25, 3 (July 2006), 805–811.
- Doug L. James, Jernej Barbič, and Dinesh K. Pai. 2006. Precomputed Acoustic Transfer: Output-sensitive, Accurate Sound Generation for Geometrically Complex Vibration Sources. *ACM Trans. Graph.* 25, 3 (July 2006), 987–995.
- Doug L. James and Dinesh K. Pai. 1999. ArtDefo: accurate real time deformable objects. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 65–72.
- Harold Jeffreys and Bertha Swirles Jeffreys. 1966. *Methods of mathematical physics* (3rd ed.). London: Cambridge University Press.
- Stefan Jeschke, Tomáš Skřivan, Matthias Müller-Fischer, Nuttapong Chentanez, Miles Macklin, and Chris Wojtan. 2018. Water surface wavelets. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 94.
- Stefan Jeschke and Chris Wojtan. 2015. Water Wave Animation via Wavefront Parameter Interpolation. *ACM Transactions on Graphics* 34, 3 (2015), 1–14.
- Stefan Jeschke and Chris Wojtan. 2017. Water Wave Packets. *ACM Transactions on Graphics (SIGGRAPH 2017)* 36, 4 (2017).
- Michael Kass and Gavin Miller. 1990. Rapid, stable fluid dynamics for computer graphics. In *ACM Siggraph Computer Graphics*, Vol. 24. ACM, 49–57.
- T. Keeler and R. Bridson. 2014. Ocean Waves Animation Using Boundary Integral Equations and Explicit Mesh Tracking. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '14)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 11–19.
- Theodore Kim, Jerry Tessendorf, and Nils Thürey. 2013. Closest Point Turbulence for Liquid Surfaces. *ACM Trans. Graph.* 32, 2, Article 15 (April 2013), 13 pages.
- Stefan Kurz, Oliver Rain, and Sergej Rjasanow. 2002. The adaptive cross-approximation technique for the 3D boundary-element method. *IEEE transactions on Magnetics* 38, 2 (2002), 421–424.
- Bernard Le Méhauté. 1988. Gravity-capillary rings generated by water drops. *Journal of Fluid Mechanics* 197 (1988), 415–427.
- Seongkyu Lee. 2017. Review: The Use of Equivalent Source Method in Computational Acoustics. *Journal of Computational Acoustics* 25, 01 (2017).
- Seongkyu Lee, Kenneth Steven Brentner, and Philip John Morris. 2010. Acoustic scattering in the time domain using an equivalent source method. *AIAA Journal* 48, 12 (1 12 2010), 2772–2780.
- Jörn Lovisich. 2002. A Convolution-Based Algorithm for Animated Water Waves. In *Eurographics 2002 - Short Presentations*. Eurographics Association.
- G. A. Mastin, P. A. Watterberg, and J. F. Mareda. 1987. Fourier Synthesis of Ocean Scenes. *IEEE Computer Graphics and Applications* 7, 3 (March 1987), 16–23.
- R. Mehra and D. Manocha. 2014. Wave-based sound propagation for VR applications. In *2014 IEEE VR Workshop: Sonic Interaction in Virtual Environments (SIVE)*. 41–46.
- Ravish Mehra, Nikunj Raghuvanshi, Lakulish Antani, Anish Chandak, Sean Curtis, and Dinesh Manocha. 2013. Wave-based Sound Propagation in Large Open Scenes Using an Equivalent Source Formulation. *ACM Trans. Graph.* 32, 2, Article 19 (2013).
- Olivier Mercier, Cynthia Beauchemin, Nils Thürey, Theodore Kim, and Derek Nowrouzezahrai. 2015. Surface Turbulence for Particle-based Liquid Simulations. *ACM Trans. Graph.* 34, 6, Article 202 (Oct. 2015), 10 pages.
- Tobias Pfaff, Nils Thürey, and Markus Gross. 2012. Lagrangian vortex sheets for animating fluids. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 112.
- Stefan A Sauter and Christoph Schwab. 2010. Boundary element methods. In *Boundary Element Methods*. Springer, 183–287.
- Jerry Tessendorf. 2004a. Interactive water surfaces. *Game Programming Gems* 4 (2004).
- Jerry Tessendorf. 2004b. *Simulating Ocean Water*. (2004).
- Jerry Tessendorf. 2014. eWave: Using an Exponential Solver on the iWave Problem. *Technical Note* (2014).
- William "Lord Kelvin" Thomson. 1891. *Popular lectures and addresses*. Vol. 3. Macmillan London. 481–8 pages.
- Nils Thürey, Ulrich Rüdte, and Marc Stamminger. 2006. Animation of Open Water Phenomena with coupled Shallow Water and Free Surface Simulations. 157–164.
- Nils Thürey, Chris Wojtan, Markus Gross, and Greg Turk. 2010. A multiscale approach to mesh-based surface tension flows. *ACM Transactions on Graphics (TOG)* 29, 4 (2010), 48.
- Steffen Weißmann and Ulrich Pinkall. 2010. Filament-based smoke with vortex shedding and variational reconnection. In *ACM Transactions on Graphics (TOG)*, Vol. 29. ACM.
- Sheng Yang, Xiaowei He, Huamin Wang, Sheng Li, Guoping Wang, Enhua Wu, and Kun Zhou. 2016. Enriching SPH Simulation by Approximate Capillary Waves. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '16)*. Eurographics Association, Goslar Germany, Germany, 29–36.
- Jihun Yu, Chris Wojtan, Greg Turk, and Chee Yap. 2012. Explicit mesh surfaces for particle based fluids. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library.
- Cem Yuksel, Donald H House, and John Keyser. 2007. Wave particles. In *ACM Transactions on Graphics (TOG)*, Vol. 26. ACM, 99.
- Yufeng Zhu, Robert Bridson, and Chen Greif. 2015. Simulating rigid body fracture with surface meshes. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 150.